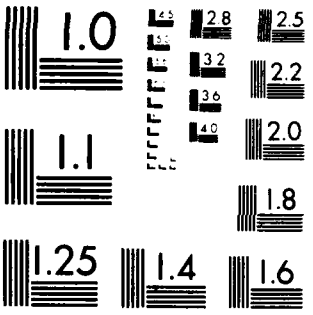
AD-A135 105 ON ACYCLIC DATABASE DECOMPOSITIONS(U) STANFORD UNIV CA 1/1
DEPT OF COMPUTER SCIENCE C BEERI ET AL. JUL 83
STAN-CS-83-976 AFOSR-TR-83-0959 AFOSR-80-0212

UNCLASSIFIED F/G 9/2 NL

END
DATE FILMED
12-83
DTIC



July 1983

Report No. STAN-CS-83-976

On Acyclic Database Decompositions

by

Catriel Beeri and Moshe Vardi

Department of Computer Science

Stanford University
Stanford, CA 94305

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 83-0959	2. GOVT ACCESSION NO. AD-A135 105	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ON ACYCLIC DATABASE DECOMPOSITIONS		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Catriel Beeri and Moshe Vardi		8. CONTRACT OR GRANT NUMBER(s) AFOSR-80-0212
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science Stanford University Stanford CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Mathematical & Information Sciences Directorate Air Force Office of Scientific Research /NM Bolling AFB DC 20332		12. REPORT DATE JUL 83 13. NUMBER OF PAGES 10
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Given a universal relation scheme, presented as a set of attributes and a set of dependencies, it may be advantageous to decompose it into a collection of schemes, each with its own sets of attributes and dependencies, which has some desired properties. A basic requirement for such a decomposition to be useful is that the corresponding decomposition map on universal relations be injective. A central problem in database theory is to find the reconstruction map, i.e., the inverse map of an injective decomposition map. The authors prove here that when the decomposition, viewed as a hypergraph, is acyclic and (CONTINUED)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ITEM #20, CONTINUED: the given dependencies are full implicational dependencies, then the reconstruction map is the natural join. Based on this, the authors show that there is a polynomial time algorithm to test for injectiveness of decompositions.

ON ACYCLIC DATABASE DECOMPOSITIONS

Catriel Beeri†

Aiken Computation Laboratory

Harvard University

Cambridge, MA 02138

Moshe Y. Vardi‡

Department of Computer Science

Stanford University

Stanford, CA 94305

July 1983

Abstract

Given a universal relation scheme, presented as a set of attributes and a set of dependencies, it may be advantageous to *decompose* it into a collection of schemes, each with its own sets of attributes and dependencies, which has some desired properties. A basic requirement for such a decomposition to be useful is that the corresponding decomposition map on universal relations be *injective*. A central problem in database theory is to find the *reconstruction* map, i.e., the inverse map of an injective decomposition map. We prove here that when the decomposition, viewed as a hypergraph, is *acyclic* and the given dependencies are *full implicational dependencies*, then the reconstruction map is the *natural join*. Based on this, we show that there is a polynomial time algorithm to test for injectiveness of decompositions.

† On leave from The Hebrew University of Jerusalem, Israel.
‡ Research supported by a Weizmann Post-Doctoral Fellowship and AFOSR - 80-0212

1. Introduction

A significant portion of research on relational database theory has been concerned with the properties of *database decompositions*. The generic problem can be described as follows. Given a "universal" scheme presented as a set of attributes and a set of dependencies, what are the conditions under which it can be decomposed into a collection of schemes, each with its own sets of attributes and dependencies, having some desired properties. The properties considered were, at first, various *normal forms*. (see [Ma, Ul]). It was then realized, however, that there is a more fundamental property, called *faithfulness*, that has to be satisfied by decompositions [R1,BR,MMSU]. Intuitively, a decomposition is *faithful* if the relations corresponding to the different schemes can be updated separately.

A basic assumption underlying these ideas is that when a universal scheme is decomposed into smaller schemes, each of the universal relations associated with it is decomposed into smaller relations using the *projection* operation, i.e., each such relation is projected onto each one of the smaller schemes. For a decomposition to be faithful, we must not lose any information by decomposing the universal relations, that is the decomposition map must be *injective*. In other words, it should be possible to reconstruct the universal relations from their projections. The desirability of injectiveness is called in [BBG] the *representation principle*.

The question raised now is as follows: Given that a decomposition is faithful, what is the resulting reconstruction map? The most natural choice for this map is the *(natural) join* operation. The problem is whether indeed the reconstruction map is the join. This problem was first presented in [R1], where it is answered affirmatively for the case that only functional dependencies are given and the decomposition is into two schemes only. This result was generalized in [BR, MMSU], where the restriction on the number of schemes was removed. It was generalized further in [V2] to the case where *full implicational dependencies* are given. In contrast, it is shown there that if we allow arbitrary first-order sentences instead of full implicational dependencies, then the reconstruction map does not have to be the join.

The result in [V2] assumes that both finite and infinite databases are considered. It is more realistic, however, to assume that database are inherently finite. This is the case that we consider in this paper. While we cannot show that the reconstruction map is the join, we prove it for the important case that the decomposition is *acyclic* [FMU, BFMY]. Furthermore, in this case the condition of injectiveness is equivalent to the condition that the reconstruction map be the join. (The same result was shown independently in [CP] for the less general case where functional and join dependencies are given and and the decomposition is into two schemes.) In contrast, it is shown in [V2] that for cyclic decomposition injectiveness does not imply that the reconstruction map is the join. Finally, based on this characterization of injectiveness, we show that there is a polynomial time algorithm to test for injectiveness of decompositions.

2. Basic Definitions

2.1. Relation and Dependencies

We assume familiarity with the terminology and concepts of relational database theory as presented in [Ma, Ul]. We use $I[X]$ to denote the *projection* of the relation I on the attribute set X, and $*_j I_j$ to denote the *join* of the set $\{I_j\}$ of relations. We assume that the relations we are dealing with and, accordingly, the dependencies that refer to them are *typed*, that is, distinct attributes have disjoint domains. We also assume that all relations are finite. The *universal relation scheme* is denoted U, and a *database scheme* is a set $\mathbf{R}=\{R_1, \ldots, R_k\}$ of distinct relation scheme such that $\bigcup_{i=1}^{k} R_i = U$.

The dependencies we use here are the *full implicational dependencies* (fid's) [BV, F2], i.e., *equality-generating dependencies* (egd's) and *full tuple-generating dependencies* (ftgd's). We denote the class of relations that satisfy a set Σ of dependencies by $SAT(\Sigma)$, and we denote logical implication by $\models$.

The class of *join dependencies* [ABU,R2] is a subclass of the class of ftgd's. They were originally introduced using the following notation. A join dependency (jd) is a statement $*[\mathbf{R}]$, where $\mathbf{R}$ is a database scheme $\{R_1, \ldots, R_k\}$. It is satisfied by a relation I on U if $I = \overset{k}{\underset{i=1}{*}} I[R_i]$. Let $\mathbf{R}$ and $\mathbf{S}$ be database schemes. We say that $\mathbf{S}$ *covers* $\mathbf{R}$ if for all R in $\mathbf{R}$ there is some S in $\mathbf{S}$ such that $R \subseteq S$. It is known [BMSU] that if $\mathbf{S}$ covers $\mathbf{R}$ then $*[\mathbf{R}] \models *[\mathbf{S}]$. A *multivalued dependency* (mvd) [F1, Za] is essentially a "binary" jd, i.e., it is a jd $*[R_1, R_2]$. (Note that $R_1 \cup R_2 = U$). This notation differs from the standard arrow notations for mvd's.

Let $\mathbf{R} = \{R_1, \ldots, R_k\}$ be a database scheme. If $\mathbf{R}_1$ and $\mathbf{R}_2$ are subsets of $\mathbf{R}$, then $\mathbf{R}_1, \mathbf{R}_2$ is a partition of $\mathbf{R}$ if $\mathbf{R}_1 \cap \mathbf{R}_2 = \emptyset$ and $\mathbf{R} = \mathbf{R}_1 \cup \mathbf{R}_2$. A database scheme $\mathbf{R}$ is said to be *acyclic* [FMU,BFMY] if the jd $*[\mathbf{R}]$ is logically equivalent to the set of mvd's

$$\{*[\bigcup \mathbf{R}_1, \bigcup \mathbf{R}_2] : \mathbf{R}_1, \mathbf{R}_2 \text{ is a partition of } \mathbf{R}\}.$$

Note that one direction of the equivalence is true for any database scheme.

We will use here two properties of fid's that are shown in [V1,V3].

(1) Let Σ be a set of fid's and let τ be an mvd. If $\Sigma \not\models \tau$ then there is a relation I such that $|I| = 2$, I satisfies Σ, and I does not satisfy τ.

(2) There is a quadratic time algorithm to test for a given finite set Σ of fid's and an mvd τ whether $\Sigma \models \tau$.

2.2. Decompositions

A database on the database $\mathbf{R} = \{R_1, \ldots, R_k\}$ is a collection $\{I_1, \ldots, I_k\}$ of relations on $R_1, \ldots, R_k$, correspondingly. With each database scheme $\mathbf{R}$ we associate a *decomposition map* $\Delta_{\mathbf{R}}$. Given a relation I on U, $\Delta_{\mathbf{R}}$ applied to I yields a database on $\mathbf{R}$:

$$\Delta_{\mathbf{R}}(I) = \{I[R_1], \ldots, I[R_k]\}.$$

A decomposition map $\Delta_{\mathbf{R}}$ is *injective* with respect to a set Σ of dependencies if $\Delta_{\mathbf{R}}$ is injective on $SAT(\Sigma)$. That is, if I and J are two distinct relations in $SAT(\Sigma)$, then $\Delta_{\mathbf{R}}(I) \neq \Delta_{\mathbf{R}}(J)$. If

Δ_R is injective with respect to Σ then it has an inverse map

$$\rho_R:\{\Delta_R(I): I \in SAT(\Sigma)\} \rightarrow SAT(\Sigma),$$

such that $\rho_R(\Delta_R(I))=I$ for all I in $SAT(\Sigma)$. ρ_R is called the *reconstruction map.* If $\rho_R(I_1, \ldots, I_k)=\overset{k}{\underset{j=1}{*}} I_j$ then we say that the reconstruction map is the join. Another map associated with a database scheme R is the *project-join map* m_R defined by

$$m_R(I)=\overset{k}{\underset{j=1}{*}} I[R_j].$$

Note that a relation I satisfies *[R] exactly when $m_R(I)=I$.

3. The Main Result

Let I be a relation on U. A *permutation* on I is an injective map α from the set of values in I into itself such that the set of values for each attribute is mapped into itself. A permutation on I is essentially a vector of permutations, one for each attribute of U. We denote by $\alpha(I)$ the relation obtained by replacing simultaneously each value in I by its image under α.

Lemma 1. [BV] Let I be a relation on U and let α be a permutation on I. Then I and $\alpha(I)$ satisfy exactly the same fid's. ■

Obviously, I and $\alpha(I)$ have the same projection on each singleton attribute set $\{A\}$. They need not have, however, the same projections on larger attribute sets. A permutation α on a relation I *preserves* a database scheme R if $\Delta_R(I)=\Delta_R(\alpha(I))$.

Lemma 2. Let $R=\{R_1,R_2\}$ be a database scheme, and let $I=\{w_1,w_2\}$ be a relation on U. For every tuple w in $m_R(I)$ there exists a permutation α on I that preserves R such that w is in $\alpha(I)$.

Proof. If w is in I then take α to be the identity permutation, so we can assume that w is not in I. If however either $R_1 \subseteq R_2$ or $R_2 \subseteq R_1$, then $m_R(I)=I$, so assume that the two sets are incomparable. We can also assume that $w_1[R_1] \neq w_2[R_1]$ and $w_1[R_2] \neq w_2[R_2]$, otherwise

$m_R(I)=I$. Finally, $w_1[R_1 \cap R_2]=w_2[R_1 \cap R_2]$, since otherwise $m_R(I)=I$.

Let $w[R_1]=w_1[R_1]$ and $w[R_2]=w_2[R_2]$. Define α to be the identity on each attribute in R_1. For an attribute in R_2, α exchanges the values $w_1[A]$ and $w_2[A]$. α is well defined, since $w_1[R_1 \cap R_2]=w_2[R_1 \cap R_2]$. Now we have that $\alpha(w_1)[R_1]=w_1[R_1]$ and $\alpha(w_1)[R_2]=w_2[R_2]$, so $w=\alpha(w_1) \in \alpha(I)$. It is also easy to see that α preserves R. ■

The relationship between covering and preservation is pointed to in the following easy lemma.

Lemma 3. Let R and S be database schemes such that S covers R. If α is a permutation on a relation I that preserves S, then it also preserves R. ■

We can now prove our main result.

Theorem 1. Let Σ be a set of fid's, and let R be an acyclic database scheme. The following three conditions are equivalent:

(1) Δ_R is injective with respect to Σ.

(2) $\Sigma \models^*[R]$.

(3) ρ_R is the join.

Proof.

(1 -> 2): Suppose that Δ_R is injective with respect to Σ but $\Sigma \not\models^*[R]$. Since R is acyclic, there is a partition R_1, R_2 of R such that $\Sigma \not\models^*[S]$, where $S=\{\bigcup R_1, \bigcup R_2\}$. By property (1) of fid's, there is a relation $I=\{w_1, w_2\}$ such that I satisfies Σ but I does not satisfy $^*[S]$. Since I does not satisfy S, there is a tuple w that is in $m_S(I)$ but not in I. By Lemma 2, there is a permutation α on I such that w is in $\alpha(I)$, and α preserves S. But then we must have $I \neq \alpha(I)$, since w is not in I, and by Lemma 1, I and $\alpha(I)$ both satisfy Σ. We also have that α preserves R, because S covers R. Therefore, $\Delta_R(I)=\Delta_R(\alpha(I))$ - in contradiction to the injectiveness of Δ_R with respect to Σ.

(2 -> 3): Let I be a relation in $SAT(\Sigma)$. Since $\Sigma \models^*[R]$, $I=m_R(I)$. That is,

$$I = \overset{k}{\underset{i=1}{*}} I[R_i] = \rho_R(\Delta_R(I)),$$

where ρ_R is the join.

(3 -> 1): If ρ_R exists, then Δ_R must be injective with respect to Σ. ■

We now show that the condition of Theorem 1 can be tested efficiently.

Theorem 2. There is cubic time algorithm to test for a given finite set Σ of fid's and an acyclic database scheme R whether Δ_R is injective with respect to Σ.

Proof. By Theorem 1 it suffices to test whether $\Sigma \models *[R]$. Our strategy is to construct first a set Γ of mvd's that is logically equivalent to $*[R]$ (by acyclicity), and then to test if $\Sigma \models \tau$ for each τ in Γ. By property (2) of fid's,, each of the latter tests can be done in quadratic time. The crux of the proof is showing how to construct Γ such that $|\Gamma| < |U|$ and the construction can be done in quadratic time.

If R is acyclic then $|R| \leq U$ and there is a set Γ of mvd's that is logically equivalent to $*[R]$ such that $|\Gamma| < |R|$ [BFMY]. In order to construct Γ we have to construct first a *join forest* for R.[1] This join forest can be constructed in time linear in the size of R [TY]. Then every mvd in Γ can be constructed from the join forest in time linear in the size of R [BFMY]. Since $|\Gamma| < |U|$, the claim follows. ■

4. Concluding Remarks

We have shown that the reconstruction map is the join for quite a general situation, namely when the dependencies are fid's and the decomposition is acyclic. We note that most classes of dependencies treated in the literature are special cases of fid's. An exception is the class of *inclusion dependencies* [CFP]. Our results can be generalized to deal also with inclusion dependencies [KCV]. As for the restriction that the decomposition be acyclic there are arguments to the effect that most real life situations can be captured by such decompositions.

---

[1] A join forest for R is a labeled acyclic graph with the elements of R as nodes, an edge connecting R_i to R_j is

We have also shown how to test efficiently for injectiveness. We mentioned that there is another desirable property of decompositions, called *surjectiveness* [V2]. Faithful decompositions are both injective and surjective. When only functional dependencies are given there is a polynomial time test for faithful [BH, BR, MMSU]. We do not know of any effective test when fid's are given, even when the decomposition is acyclic.

References

[ABU] Aho, A.V., Beeri, C., Ullman, J.D.: The theory of joins in relational databases. ACM Trans, on Database Systems 4(1979), pp.297-314.

[BBG] Beeri, C., Bernstein, P.A., Goodman, N.: A sophisticates' introduction to database normalization theory. Proc. Int'l Conf. on Very Large Databases, Berlin, 1978, pp.113-124.

[BFMY] Beeri, C., Fagin, R., Maier, D., Yannakakis, M.: On the desirability of acyclic database schemes. IBM Research Report RJ3131 (version 2), April 82. Also, to appear in J. of ACM.

[BH] Beeri, C., Honeyman, P.: Preserving functional dependencies. SIAM J. on Comput. 10(1981), pp. 647-656.

[BMSU] Beeri, C., Mendelzon, A.O., Sagiv, Y, Ullman, J.D.: Equivalence of relational database schemes. SIAM J. on Comput. 10(1981), pp. 647-656.

[BR] Beeri, C., Rissanen, J.: Faithful representation of relational database schemes. IBM Research Report, San Jose, 1980.

[BV] Beeri, C., Vardi, M.Y.: A proof procedure for data dependencies. Department of Computer Science, The Hebrew University of Jerusalem, Dec. 1980.

[CFP] Casanova, M.A., Fagin, R., Papadimitriou, C.H.: Inclusion dependencies and their interaction with functional dependencies. 1st ACM Symp. on Principles of Database

labeled by $R_i \cap R_j$, and for each attribute A the subgraph of nodes and edges that contain A is connected.

Systems, Los Angeles, 1982, pp. 171-176.

[CP] Cosmadakis, S.S., Papadimitriou, C.H.: Updates of relational views. 2nd ACM Symp. on Principles of Database Systems, Atlanta, 1983, pp. 317-331.

[F1] Fagin, R.: Multivalued dependencies and a new normal form for relational databases. ACM Trans. on Database Systems 2(1977), pp. 262-278

[F2] Fagin, R.: Horn clauses and database dependencies. J. of ACM 29(1982) pp. 952-983.

[FMU] Fagin, R., Mendelzon, A.O., Ullman, J.D.: "A simplified universal relation assumption and its properties," ACM Trans. on Database Systems 7(1982), pp. 343-360.

[KCV] Kanellakis, P.C., Cosmadakis, S.S., Vardi, M.Y.: Unary inclusion dependencies have polynomial time inference problems, Proc. ACM Symp. on Theory of Computing, Boston, April 1983, pp. 264-277.

[Ma] Maier, D., The Theory of Relational Databases, Computer Science Press, Rockville, Maryland, 1983.

[MMSU] Maier, D., Mendelzon, A.O., Sadri, F., Ullman, J.D.: Adequacy of decompositions of relational databases. In *Advances in Database Theory* (H. Gallaire, J. Minker, and J.M. Nicolas, eds.), Plenum Press, 1981, pp. 101-114.

[R1] Rissanen, J.: Independent components of relations. ACM Trans. on Database Systems 2(1977), pp. 317-325.

[R2] Rissanen, J.: Theory of relations for databases - a tutorial survey. Proc. 7th Symp. on Math. Found. of Computer Science, 1978, in Lecture Notes in Computer Science 64, Springer-Verlag, Berlin, pp. 537-551.

[TY] Tarjan, R.E., Yannakakis, M.: Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. Bell Laboratories Technical Report, 1982.

[Ul] Ullman, J. D., Principles of Database Systems, Computer Science Press, Potomac, Maryland, 1983.

[V1] Vardi, M.Y.: The implication problem for data dependencies in the relational model, Ph.D. Dissertation (in Hebrew), Dept. of Computer Science, The Hebrew University of Jerusalem, Sept. 1981.

[V2] Vardi, M.Y.: On the decomposition of relational databases. Proc. IEEE Symp. on Foundation of Computer Science, Chicago, Nov. 1982, pp. 176-187.

[V3] Vardi, M.Y.: Inferring multivalued dependencies from full implicational dependencies. To appear.

[Za] Zaniolo, C.: Analysis and design of relational schemata for database systems. Technical Report UCLA-ENG-7769, UCLA, 1976.


FIL